DOCUMENT RESUME

ED 460 006 SE 065 498

AUTHOR	Wieschenberg, Agnes A.
TITLE	Logic via Computer Programming.
PUB DATE	1999-11-00
NOTE	6p.; Paper presented at the Annual International Conference on Technology in Collegiate Mathematics (12th, San Francisco, CA, November 4-7, 1999).
AVAILABLE FROM	For full text: http://archives.math.utk.edu/ICTCM/EP-12/C27/pdf/paper.pdf.
PUB TYPE	Opinion Papers (120) -- Speeches/Meeting Papers (150)
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	*Algorithms; *Computer Uses in Education; Higher Education; *Mathematics; Programming; Teaching Methods; *Thinking Skills

ABSTRACT

This paper proposed the question "How do we teach logical thinking and sophisticated mathematics to unsophisticated college students?" One answer among many is through the writing of computer programs. The writing of computer algorithms is mathematical problem solving and logic in disguise and it may attract students who would otherwise stop taking mathematics courses after their required sequence is finished. In college classrooms in the United States, there is often an over-involvement with the calculation aspect of mathematics, especially in today's technical environment. The emphasis should fall on the teachers' developing of logic in students. Just like mathematical algorithms, computer algorithms however simple, employ logical steps which will result in the desired conclusion. Mathematics teachers should take advantage of the inumerable opportunities, even in a beginner's computer programming course, to play with algorithms that may aid students in the development of logical ways to approach mathematical problems. (MA)

Reproductions supplied by EDRS are the best that can be made
from the original document.

PERMISSION TO REPRODUCE AND DISSEMINATE THIS MATERIAL HAS BEEN GRANTED BY
A. Wieschenberg
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☒ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

Agnes A. Wieschenberg
John Jay College of Criminal Justice
Department of Mathematics
445 West 59th Street, New York, N.Y. 10019
phone:(212)237-8924, fax:(201)567-1811, E-mail awieschen@aol.com

ED 460 006

LOGIC VIA COMPUTER PROGRAMMING

How do we teach logical thinking and sophisticated mathematics to unsophisticated college students? There are many answers to the proposed question, one of them -- through writing computer programs. The writing of computer algorithms is mathematical problem solving and logic in disguise and it may attract students who would otherwise stop taking mathematics courses after their required sequence is finished.

Looking back at the history of computing, we discover that it is an ancient activity with roots dating back to the Babylonian, Egyptian and Greek civilizations. The motivation for the above activity has always been the search for accurate and efficient computation with the least effort on the side of the person responsible for finding the result. This idea is best expressed by Gottfried Wilhelm Leibnitz, seventeenth century mathematician (1646-1716) who wrote: *"It is unworthy of excellent [persons] to lose hours like slaves in the labor of calculations which could be safely relegated to anyone else if machines were used."* In college classrooms in the United States we are sometimes overly involved with the calculation aspect of mathematics when, especially in todays technical environment, the emphases should fall on the developing of logic in our students.

In mathematics we deal with numerous algorithmic problems, problems that can be solved by following a set of executable instructions. Just like mathematical algorithms, computer algorithms however simple, employ logical steps which will result in the desired conclusion. In particular the use

SE 058 498

BEST COPY AVAILABLE

of the "if" structure and the recursive nature of the "for" and "while" loops will help students to develop logical thinking. The rewards in using the computer are immediate. Running the program that the student has just written will allow him/her to "see" if it has worked or not. More often than not, the program will contain logical errors and debugging the program will allow the student to think about the logical flow of the program again and find where the problem occurred. (The computer will not pick up logical errors!) Students have an unlimited number of opportunities to correct the algorithm and execute the program over and over again until the error is discovered at which point a new concept is learned. Chances are that the same mistake will not be made at a following example where the same or a similar technique could be employed.

Some programs run and produce the desired results but, as mathematician sometimes say, it is not an "elegant" solution. Students can then be encouraged to use algorithms that will be best fitted in a particular set of circumstances. The best students may be asked to find the algorithm that leads to the least number of operations performed by the computer.

Let us look at the following compound interest problem. Students are normally given the formula

[1] $a = p(1 + r)^n$

where p is the principal (original amount invested)
r is the annual interest rate
n is the number of years
a is the amount of deposit at the end of the *n*th year

If the student is asked to print out the amount on deposit at the end of the *n*th year that can be accomplished by simply coding the above formula [1] and getting the result immediately. Chances are that the majority of the students have no idea where the formula came from. If on the other hand

BEST COPY AVAILABLE

we ask the student to print out the amount each year, including the final amount, the student will have to employ a loop structure. Most likely than not, he/she will code the following loop given initial deposit of $1000 at the annual rate of 10% for 10 years:

[2]
```
#include<iostream.h>
#include<iomanip.h>
#include<math.h>
main()
{
int n;    //number of years on deposit
double r = 0.1,   //annual rate of interest
 p = 1000.0,    //original amount principal
 a;    //amount on deposit
cout<<"YEAR"<<setw(20)<<"AMOUNT"<<endl<<endl;
for ( n = 1; n <= 10; n++) {
     a = p * pow( 1 + r , n);
      cout<<setw(4)<<n
           <<setiosflags(ios::fixed | ios::showpoint)
           <<setprecision(2)<<setw(20)<<a<<endl;
  }
 return 0;
  }
```

This program works well and the desired result will be calculated and printed. We get accurate information but many students probably don't understand how the above formula works. At this point, the following assignment is given: in 1626 Manhattan Island was sold by its Indian owners to a Dutch man, by the name of Peter Minuit, for $24.00. If this money would have been placed on deposit in a bank at 4% annual interest rate, how much would be on deposit at the end of 1999 ? Programs that were handed back to me by my students used the same analogy as we did in program [2].

[3]
```
for ( n = 1; n<=373; n++ )
    a = 24.00*pow( 1 + 0.04, n );
```

This means that, for example, to get the 371st year's amount on deposit the computer has to compute (1 + 0.04), take that to the 371st power (even though $(1+0.04)^{370}$ is in memory) and multiply the result

by 24.0, resulting in three computations, an addition, a multiplication and an exponantiation. The last operation, exponantiation, is not machine efficient.

[4] $a = 24.0(1+0.04)^{371}$ or coded a = 24.0 *pow(1+0.04, 371)

which results in a = 50,070,291.70

Since the computer already has a figure stored for $(1 + 0.04)^{370}$, (48,144,511.25), the calculation becomes a redundant effort each time the loop executes.

There is an algorithmic approach to this program which uses the recursive nature of the formula using results that are already known. It saves calculations and computing time. It requires only one addition and one multiplication each time the loop executes:

[5] new principal = old principal (1+rate)

new principal = 48,144,511.25(1+0.04) = 50,070,291.70

In an assignment statement this can be expressed as

[6] p = p*(1 + r);

Placed in a loop, this assignment statement will produce the same result by using each new yearly value and only multiplying each time the loop goes around, by (1+r) in each loop, rather than raising the expression (1 + r) to the1,2,3,...372, 373rd powers in each loop as the computer had to do in the expression,

a = p*pow(1 + r , n);

as used in examples [2] and[4]. Understanding the difference in how the two different programs will produce the results will hopefully generate a deeper understanding of the mathematics involved and therefore promote the logical process.

Another interesting way to test and promote logical thinking may be the following fun example that often appears in my exams. It has to do with the ticketing of drivers who exceed the speed limit:

[7]
```
if ( speed > 35 )
   cout<<"Issue ticket at  $40.00"<<endl;
else if ( speed > 50 )
    cout<<"Issue ticket at 60.00"<<endl;
else if ( speed > 80 )
     cout<<"Issue ticket at $100.00"<<endl;
else
     cout<<"No ticket issued"<<endl;
```

Intuitively, when students are asked how much does a driver have to pay if caught speeding at 85 miles/hour they most likely will reply $100.00. This is what our brain tells us. The faster you go, the more you pay. (I often warn my students that when they are checking their program they should not do what logic would dictate, rather they should put themselves in place of the computer and blindly follow the instructions just as the computer would.) Just as in the above example if we look at the logical flow of the program, we quickly realize that every speeder will pay only $40.00 regardless of speed. Only the driver who drives 35 miles/hour or less will get away without getting a ticket. Once we point out the faulty logic used in program [7] most students will fully understand the problem and will be able to correct the algorithm.

The time allotted for this presentation does not allow me to give more examples but there are plenty of opportunities, even in a beginner's computer programming course, to play with algorithms that may aid students in the development of logical ways to approach mathematical problems.